

TUGAS BESAR 2 IF3170 INTELIGENSI BUATAN
Implementasi Algoritma Pembelajaran Mesin pada Pemrosesan Data



Disusun oleh:

13521057 Hosea Nathanael Abetnego
13521059 Arleen Chrysanthia Gunardi
13521127 Marcel Ryan Antony
13521145 Kenneth Dave Bahana

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

Daftar Isi.....	i
Daftar Tabel.....	ii
Daftar Gambar.....	iii
1. Implementasi KNN.....	1
2. Implementasi Naive-Bayes.....	4
3. Perbandingan Hasil Prediksi dari Algoritma yang Diimplementasikan dengan Hasil yang Didapatkan dengan Menggunakan Pustaka.....	9
4. Kaggle.....	13
5. Referensi.....	15
6. Pembagian Pekerjaan.....	16
7. Lampiran.....	16

Daftar Tabel

Tabel 1.1 Fungsi-Fungsi pada Algoritma K-Nearest Neighbor.....	3
Tabel 2.1 Fungsi-Fungsi pada Algoritma Naive-Bayes.....	6
Tabel 3.1 Perbandingan Algoritma KNN yang Diimplementasikan dengan Pustaka.....	10
Tabel 3.2 Perbandingan Algoritma Naive-Bayes yang Diimplementasikan dengan Pustaka.....	11
Tabel 4.1 Pembersihan Data.....	14
Tabel 6.1 Pembagian Pekerjaan.....	16

Daftar Gambar

Gambar 1.1 Rumus Euclidean Distance.....	1
Gambar 1.2 Contoh Data Perhitungan Jarak.....	2
Gambar 2.1 Rumus untuk Menentukan Peluang $P(A B)$	4
Gambar 2.2 Contoh Data Perhitungan Prediksi.....	4
Gambar 2.3 Rumus Distribusi Normal.....	5
Gambar 2.4 Akurasi Hasil Prediksi Menggunakan Algoritma Naive-Bayes.....	9

1. Implementasi KNN

K-Nearest Neighbors merupakan salah satu algoritma *supervised learning*. Algoritma ini adalah algoritma *machine learning* yang bersifat non-parametrik dan *lazy learning*. Metode yang bersifat non-parametrik memiliki makna bahwa metode tersebut tidak membuat asumsi apa pun tentang distribusi data yang mendasarinya. Algoritma ini dikatakan *lazy learning* karena tidak menggunakan data *training* untuk membuat model.

Dalam melakukan prediksi data, pertama-tama algoritma KNN mengukur jarak nilai data tersebut terhadap seluruh instansi data yang ada di data latih yang digunakan. Kemudian, KNN akan memilih sebanyak K data yang memiliki jarak terdekat dan sejumlah K data itu akan digunakan untuk mencari mayoritas nilai target yang diinginkan. Dalam kasus data yang telah ditentukan, dari sejumlah K data akan diambil data mayoritas dengan nilai *price range* yang paling banyak. Data mayoritas tersebut yang akan dijadikan kesimpulan prediksi *price range* data tersebut.

Pengukuran jarak antar nilai data dari data latih dan uji menggunakan perumusan jarak *Euclidean*. Jarak *Euclidean* antara dua titik atau data dalam ruang n -dimensi yang pada kasus ini adalah n jumlah kolom data didefinisikan seperti pada Gambar 1.1 berikut.

$$\text{Euclidean Distance} = |X - Y| = \sqrt{\sum_{i=1}^{i=n} (x_i - y_i)^2}$$

X: Array or vector X

Y: Array or vector Y

x_i : Values of horizontal axis in the coordinate plane

y_i : Values of vertical axis in the coordinate plane

n: Number of observations

Gambar 1.1 Rumus *Euclidean Distance*

Sebagai contoh, terdapat data seperti pada Gambar 1.2 berikut.

Classify New Instance: <Sunny, Cool, High, True>

outlook	temp.	humidity	windy	play	Distance	
sunny	hot	high	false	no	2	k = 1 → D-2 → Play: No
sunny	hot	high	true	no	1	
overcast	hot	high	false	yes	3	k = 2 → D-1, D-2, → Play: No
rainy	mild	high	false	yes	3	
rainy	cool	normal	false	yes	3	k = 3 → D-1, D-2, D-6 → Play: No
rainy	cool	normal	true	no	2	
overcast	cool	normal	true	yes	2	
sunny	mild	high	false	no	2	
sunny	cool	normal	false	yes	2	
rainy	mild	normal	false	yes	4	
sunny	mild	normal	true	yes	2	
overcast	mild	high	true	yes	2	
overcast	hot	normal	false	yes	4	
rainy	mild	high	true	no	2	

Gambar 1.2 Contoh Data Perhitungan Jarak

Sumber: Materi Kuliah *Supervised Learning K-Nearest Neighbour*, Edunex IF3170 Inteligensi Buatan Teknik Informatika ITB 2023

Dari data tersebut, kolom *play* merupakan kolom target dan sekumpulan target lainnya yang berupa atribut non-numerik digunakan untuk pengukuran jarak (untuk kasus ini, jarak *euclidean* data cukup dengan perbedaan nilai kategori sebagai 1). Dari data tersebut, dapat dilihat bahwa setiap perbedaan mempengaruhi nilai *distance* bertambah 1 untuk setiap perbedaannya. Untuk pemilihan $k = 1$, jarak terkecil yang ada pada data terletak pada data baris ke-2, yaitu bernilai 1. Pada data tersebut, nilai pada kolom *play* adalah “no” sehingga dapat disimpulkan instansi baru diklasifikan sebagai “no”. Untuk $k = 2$, nilai pada kolom *play* juga bernilai “no”. Untuk $k = 3$, dipilih data pertama, kedua, dan keenam. Dari ketiga data tersebut juga, mayoritas nilai pada kolom *play* bernilai “no”.

Untuk pemrosesan data yang diberikan pada tugas besar ini, diimplementasikan algoritma K-Nearest Neighbors. Berikut merupakan implementasi dari algoritma KNN yang telah dijelaskan.

Tabel 1.1 Fungsi-Fungsi pada Algoritma K-Nearest Neighbor

knn.py
<pre>def neighbour(train_data, test_instance, k): # Iteration along axis (all data trains) to calculate euclidean distance between test instance and the whole train data distances = np.apply_along_axis(lambda x: euclidean_distance(x, test_instance), 1, train_data) # Sorting all distances to collect the shortest distances with k amount. sorted_indices = np.argsort(distances) return sorted_indices[:k]</pre>
<pre>def knn(data, test_instance, k): # Get target attribute price_range = data['price_range'].values # Data train without the target attribute data_train = data.drop('price_range', axis=1).values # Getting k amount of shortest distance values (just the index) neighbours_idx = neighbour(data_train, test_instance, k) neighbour_price_range = price_range[neighbours_idx] # returning mode of most frequent price range value collected from neighbors. return pd.Series(neighbour_price_range).mode().iloc[0]</pre>
<pre>def knn_prediction(data_train, data_validation, k): # Iteration for multiple instances predictions = [] for i in range(len(data_validation)): test_instance = data_validation.iloc[i, :].values predictions.append(knn(data_train, test_instance, k)) return np.array(predictions)</pre>

2. Implementasi Naive-Bayes

Naive-Bayes adalah salah satu algoritma klasifikasi yang memanfaatkan teorema Bayes dengan asumsi bahwa setiap atribut yang digunakan untuk memprediksi tidak ada hubungan satu dengan yang lainnya. Untuk atribut numerik dan non-numerik, algoritma ini memiliki perhitungan yang berbeda.

Untuk melakukan prediksi menggunakan atribut non-numerik, digunakan rumus yaitu $P(A|B)$ dari teorema Bayes dengan A adalah klasifikasi target dan B adalah nilai dari atribut. Berikut merupakan rumus peluang teorema Bayes seperti yang ditunjukkan pada Gambar 2.1 berikut.

$$\begin{aligned} P(A \cap B) &= P(A|B).P(B) = P(B|A).P(A) \\ &= P(A|B) = \frac{P(B|A).P(A)}{P(B)} \end{aligned}$$

Gambar 2.1 Rumus untuk Menentukan Peluang $P(A|B)$

Sebagai contoh, terdapat data seperti pada Gambar 2.2 berikut:

Gender	Illness
Male	No
Female	Yes
Male	Yes
Male	No
Female	No
Female	No
Male	Yes
Male	No

Frequency / Likelihood Table		Illness(heart disease)		
		Yes	No	
Gender	Male	2/5	3/5	5/8
	Female	1/3	2/3	3/8
		3/8	5/8	

Gambar 2.2 Contoh Data Perhitungan Prediksi

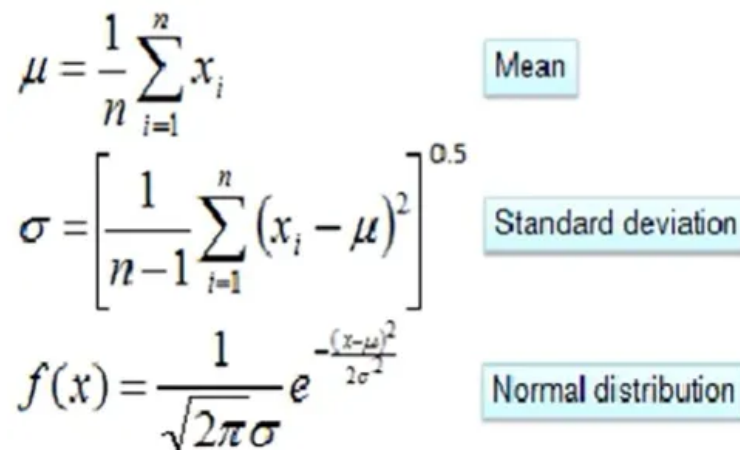
Kolom *illness* merupakan kolom target dan atribut *gender* adalah atribut yang digunakan untuk melakukan prediksi. Dengan demikian, $P(\text{illness}|\text{gender})$ perlu ditentukan, dengan

$P(\text{Yes}|\text{Male})$, $P(\text{Yes}|\text{Female})$, $P(\text{No}|\text{Male})$, dan $P(\text{No}|\text{Female})$. Dengan melihat tabel, dapat diperoleh peluang-peluang berikut:

- $P(\text{Male}|\text{Yes}) = 2/5$ and $P(\text{Male} | \text{No}) = \frac{3}{8}$
- $P(\text{Female}|\text{Yes}) = 1/3$ and $P(\text{Female}|\text{No}) = \frac{2}{3}$
- $P(\text{Male}) = 5/8$ and $P(\text{Female}) = \frac{3}{8}$
- $P(\text{Yes}) = 3/8$ and $P(\text{No}) = \frac{5}{8}$

Dari peluang-peluang yang telah diperoleh tersebut, maka dapat dilakukan kalkulasi untuk menentukan nilai prediksi. Diperoleh nilai-nilai yaitu, $P(\text{Yes}|\text{Male}) = 0.24$, $P(\text{Yes}|\text{Female}) = 0.33$, $P(\text{No}|\text{Male}) = 0.6$, dan $P(\text{No}|\text{Female}) = 1.11$. Nilai-nilai tersebut akan digunakan untuk memprediksi pada saat ada data baru.

Berbeda dengan atribut numerik, perhitungan perlu dilakukan dengan menggunakan rumus distribusi normal seperti pada Gambar 2.3 berikut.



$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Mean

$$\sigma = \left[\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2 \right]^{0.5}$$

Standard deviation

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normal distribution

Gambar 2.3 Rumus Distribusi Normal

Nilai x pada rumus tersebut merupakan nilai yang digunakan untuk melakukan prediksi. Nilai prediksi yang diperoleh dari perhitungan tersebut kemudian dipasangkan dengan klasifikasi target yang sudah ada sehingga diketahui hasil prediksi berdasarkan nilai yang diperoleh dari kalkulasi tersebut.

Setelah mengetahui tentang bagaimana algoritma Naive-Bayes bekerja, berikut adalah implementasi dari algoritma Naive-Bayes untuk diterapkan pada pemrosesan data yang diberikan pada tugas besar ini.

Tabel 2.1 Fungsi-Fungsi pada Algoritma Naive-Bayes

naivebayes.py
<pre>def separate_feature(self, X, y): """ X : array, list of all columns except target and its value y : array, list of target column value return : Classify which rows belongs to certain y value """ # Make dictionary with y value as keys, and X as values separated_feature = {} for i in range(len(X)): column_values = X[i] feature_name = y[i] if feature_name not in separated_feature: separated_feature[feature_name] = [] separated_feature[feature_name].append(column_values) return separated_feature</pre>
<pre>def statistics_info(self, X): """ Calculates mean and standard deviation for each column X : the data to be calculated return : mean and standard deviation of each column """ for column in zip(*X): # Unpack the data first and then search for every column std and mean yield { 'std' : np.std(column), 'mean' : np.mean(column), }</pre>
<pre>def fit(self, X, y): """ Training the model</pre>

```
X : array, list of all columns except target and its value
y : array, list of target column value
```

Returns : Dictionary containing prior probability (probability of getting a certain target value without any evidence), mean, and std of every column

```
"""
separated_features = self.separate_feature(X, y)
self.column_info = {}

for feature_name, column_values in separated_features.items():
    self.column_info[feature_name] = {
        "prior_probability" : len(column_values)/len(X),
        "statistics" : [i for i in self.statistics_info(column_values)],
    }
return self.column_info
```

```
def gaussian_distribution(self, x, mean, std):
    """
    Calculation of predicted values using the Gaussian distribution formula on one
    of the attributes
    x : variable value
    mean : the average attribute value of the variable
    std : standard deviation of the variable attribute

    return : predicted value
    """
    exponent = np.exp(-((x - mean)**2 / (2 * std**2)))
    return exponent/(std * np.sqrt(2*np.pi))
```

```
def predict(self, X):
    """
    Prediction calculations using all attributes
    X : data used to make predictions

    return : prediction result data for each row
    """
    hypotheses = []
    for row in X:
        joint_probability = {}
        for feature_name, columns in self.column_info.items():
```

```

        total_columns = len(columns['statistics'])
        likelihood = 1

        for idx in range(total_columns):
            feature_value = row[idx]
            mean = columns['statistics'][idx]['mean']
            std = columns['statistics'][idx]['std']
            normal_probability = self.gaussian_distribution(feature_value, mean,
std)

            likelihood *= normal_probability

        prior_probability = columns['prior_probability']
        joint_probability[feature_name] = prior_probability * likelihood

        hypothesis = max(joint_probability, key=joint_probability.get)
        hypotheses.append(hypothesis)

    return hypotheses

def accuracy(self, y_test, y_pred):
    """
    Calculation of suitability of predicted values and validation values
    y_test : validation data target values
    y_pred : training data target values

    return : accuracy between training predictions and validation values
    """
    true = 0
    for y_t, y_p in zip(y_test, y_pred):
        if y_t == y_p:
            true += 1
    return true / len(y_test)

```

Dengan implementasi algoritma tersebut, dilakukan percobaan menggunakan data numerik yang telah disediakan pada spesifikasi tugas besar, data non-numerik tidak digunakan sebab saat dilakukan percobaan dengan menggunakan pustaka tidak memiliki faktor yang signifikan terhadap hasil prediksi. Hasil akurasi prediksi menggunakan algoritma tersebut adalah sebagai berikut

```
model = NaiveBayesClassifier()
✓ 0.0s

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# Menghitung accuracy implementasi NaiveBayes
accuracy = accuracy_score(y_test, y_pred)
print("NaiveBayesClassifier accuracy: {0:.9f}".format(accuracy))

# Menghitung precision implementasi NaiveBayes
precision = precision_score(y_test, y_pred, average='micro')
print("NaiveBayesClassifier precision: {0:.9f}".format(precision))

# Menghitung recall implementasi NaiveBayes
recall = recall_score(y_test, y_pred, average='micro')
print("NaiveBayesClassifier recall: {0:.9f}".format(recall))
✓ 0.1s

NaiveBayesClassifier accuracy: 0.801335559
NaiveBayesClassifier precision: 0.801335559
NaiveBayesClassifier recall: 0.801335559
```

Gambar 2.4 Akurasi Hasil Prediksi Menggunakan Algoritma Naive-Bayes

Algoritma yang sebelumnya telah dijelaskan dikemas dalam sebuah kelas yaitu `NaiveBayesClassifier` kemudian melakukan prediksi menggunakan kelas tersebut dan menghasilkan *accuracy*, *precision*, dan *recall* sebesar 0.801335559 atau sekitar 80.133% akurat.

3. Perbandingan Hasil Prediksi dari Algoritma yang Diimplementasikan dengan Hasil yang Didapatkan dengan Menggunakan Pustaka

Hasil prediksi algoritma K-Nearest Neighbors yang diimplementasikan memiliki nilai *accuracy*, *precision*, dan *recall* sebesar 0.911666667. Nilai *accuracy*, *precision*, dan *recall* ini sama dengan akurasi hasil prediksi algoritma yang didapatkan dengan menggunakan pustaka

Scikit-learn, yaitu sebesar 0.911666667. Adapun implementasi perbandingan penggunaan algoritma yang diimplementasikan dengan algoritma pada pustaka adalah sebagai berikut.

Tabel 3.1 Perbandingan Algoritma KNN yang Diimplementasikan dengan Pustaka

<pre>df = pd.read_csv('./data/data_train.csv') df_test = pd.read_csv('./data/data_validation.csv')</pre>
<pre>column_name = 'px_height' # Replace values less than 217 with 217 in the specified column using numpy df[column_name] = np.where(df[column_name] < 217, 217, df[column_name]) column_name = "sc_w" df[column_name] = np.where(df[column_name] < 2.5, 2.5, df[column_name])</pre>
<pre># Init: define X & y X_train = df.drop(columns=['m_dep', 'talk_time', 'clock_speed', 'n_cores', 'pc', 'blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi'], axis=1) y_train = df['price_range'] X_test = df_test.drop(columns=['price_range', 'm_dep', 'talk_time', 'clock_speed', 'n_cores', 'pc', 'blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi'], axis=1) y_test = df_test['price_range']</pre>
<pre>y_pred = knn_prediction(X_train, X_test, 1) # Menghitung accuracy implementasi KNN accuracy = accuracy_score(y_test, y_pred) print("Implemented KNN accuracy: {0:.9f}".format(accuracy)) # Menghitung precision implementasi KNN precision = precision_score(y_test, y_pred, average='micro') print("Implemented KNN precision: {0:.9f}".format(precision)) # Menghitung recall implementasi KNN recall = recall_score(y_test, y_pred, average='micro') print("Implemented KNN recall: {0:.9f}".format(recall))</pre>

```

X_train = X_train.drop(columns='price_range', axis=1)
y_train = df['price_range']

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

y_pred_library = knn.predict(X_test)

# Menghitung accuracy built-in library KNN
accuracy = accuracy_score(y_test, y_pred_library)
print("Scikit-learn KNeighborsClassifier accuracy:", accuracy)

# Menghitung precision built-in library KNN
precision = precision_score(y_test, y_pred_library, average='micro')
print("Scikit-learn KNeighborsClassifier precision: {0:.9f}".format(precision))

# Menghitung recall built-in library KNN
recall = recall_score(y_test, y_pred_library, average='micro')
print("Scikit-learn KNeighborsClassifier recall: {0:.9f}".format(recall))

```

Kemudian hasil prediksi algoritma Naive-Bayes, seperti yang telah dijelaskan sebelumnya, menghasilkan *precision*, *accuracy*, dan *recall* sebesar 0.801335559. Dengan menggunakan pustaka Scikit-learn, diperoleh juga *precision*, *accuracy*, dan *recall* 0.801335559. Pengujian dilakukan dengan eksekusi kode berikut:

Tabel 3.2 Perbandingan Algoritma Naive-Bayes yang Diimplementasikan dengan Pustaka

```

df = pd.read_csv('./data/data_train.csv')
df_test = pd.read_csv('./data/data_validation.csv')

column_name = 'px_height'
# Replace values less than 217 with 217 in the specified column using numpy
df[column_name] = np.where(df[column_name] < 217, 217, df[column_name])

column_name = "sc_w"
df[column_name] = np.where(df[column_name] < 2.82, 2.82, df[column_name])

dropped_columns = ['m_dep', 'talk_time', 'clock_speed', 'n_cores', 'pc', 'blue',

```

```
'four_g', 'dual_sim', 'three_g', 'touch_screen', 'wifi']
```

```
df = df.drop(columns=dropped_columns)  
df_test = df_test.drop(columns=dropped_columns)
```

```
train = df.values.tolist()  
test = df_test.values.tolist()  
  
print(train)
```

```
for i in range(1, len(train)):  
    for j in range(len(train[i])):  
        train[i][j]=float(train[i][j])  
  
for i in range(1, len(test)):  
    for j in range(len(test[i])):  
        test[i][j]=float(test[i][j])
```

```
X_train = []  
y_train = []
```

```
X_test = []  
y_test = []
```

```
for i in range(1, len(train)):  
    X_train.append(train[i][:-1])  
    y_train.append(train[i][-1])
```

```
for i in range(1, len(test)):  
    X_test.append(test[i][:-1])  
    y_test.append(test[i][-1])
```

```
model = NaiveBayesClassifier()
```

```
model.fit(X_train, y_train)  
y_pred = model.predict(X_test)
```

```
# Menghitung accuracy implementasi NaiveBayes  
accuracy = accuracy_score(y_test, y_pred)  
print("NaiveBayesClassifier accuracy: {0:.9f}".format(accuracy))
```



```

# Menghitung precision implementasi NaiveBayes
precision = precision_score(y_test, y_pred, average='micro')
print("NaiveBayesClassifier precision: {0:.9f}".format(precision))

# Menghitung recall implementasi NaiveBayes
recall = recall_score(y_test, y_pred, average='micro')
print("NaiveBayesClassifier recall: {0:.9f}".format(recall))

librarymodel = GaussianNB()
librarymodel.fit(X_train, y_train)
y_pred = librarymodel.predict(X_test)

# Menghitung accuracy built-in library NaiveBayes
accuracy = accuracy_score(y_test, y_pred)
print("Scikit-learn GaussianNB accuracy: {0:.9f}".format(accuracy))

# Menghitung precision built-in library NaiveBayes
precision = precision_score(y_test, y_pred, average='micro')
print("Scikit-learn GaussianNB precision: {0:.9f}".format(precision))

# Menghitung recall built-in library NaiveBayes
recall = recall_score(y_test, y_pred, average='micro')
print("Scikit-learn GaussianNB recall: {0:.9f}".format(recall))

```

4. Kaggle

Algoritma yang digunakan untuk memprediksi data submisi adalah algoritma KNN. Penggunaan algoritma ini juga menentukan nilai k yang tepat untuk memberikan data yang akurat. Data submisi pada Kaggle merupakan data yang sudah bersih. Proses *data cleaning* yang dilakukan adalah dengan mengganti nilai yang tidak realistis serta melakukan *filter* kolom.

Dari setiap kemungkinan nilai k yang ada, untuk mendapatkan akurasi sebaik mungkin, kami menggunakan nilai $k = 1$. Hal yang menjadi pertimbangan utama dalam pemilihan tersebut, yaitu terkait dengan data pengujian dianggap sebagai representasi dunia nyata dan jumlah data yang banyak. Data pada nyatanya tidak tentu memiliki kurtosis yang baik atau data yang merata.

Oleh karena itu, terkait anggapan tersebut dan jumlah data yang banyak, data memiliki variansi yang cukup tinggi sehingga dengan menggunakan $k = 1$ dapat memberikan akurasi yang lebih baik karena data yang semakin tersebar dengan variansi tinggi serta jumlah data yang sangat banyak akan lebih akurat dengan pengukuran satu data latih terdekat dibandingkan dengan semakin banyak pengukuran yang dapat mengurangi ketepatan prediksi berhubung dengan variansi data yang tinggi.

Pembersihan data dilakukan terhadap data-data pada kolom *px_height* dan *sc_w* karena terdapat nilai-nilai yang tidak realistis. Contohnya, pada kolom *px_height* terdapat nilai 0, padahal resolusi layar tidak mungkin 0 pixel. Hal yang sama terdapat pula pada kolom *sc_w*. Oleh karena itu, dengan mengasumsikan ukuran tinggi pixel minimal layar ponsel adalah 217 pixel, maka data yang memiliki nilai kurang dari 217 pada kolom *px_height* digantikan dengan nilai 217. Hal yang sama berlaku juga pada kolom *sc_w*. Dengan mengasumsikan ukuran lebar minimal layar ponsel adalah 2.82 inch, maka data yang memiliki nilai kurang dari 2.82 pada kolom *sc_w* digantikan dengan nilai 2.82.

Pembersihan data kemudian dilakukan dengan melakukan *filter* terhadap kolom-kolom data yang tidak memiliki korelasi tinggi dengan *target*. Adapun kolom dieliminasi merupakan kolom dengan data non-numerik, yaitu *blue*, *dual_sim*, *four_g*, *three_g*, *touch_screen*, dan *wifi*. Selain itu, kolom numerik dengan korelasi rendah terhadap kolom target (*price_range*) juga dieliminasi, yaitu *m_dep*, *talk_time*, *clock_speed*, *n_cores*, dan *pc*. Dengan demikian, data dapat dikatakan bersih.

Tabel 4.1 Pembersihan Data

```
df = pd.read_csv('./data/data_train.csv')
df_test = pd.read_csv('./data/data_validation.csv')

column_name = 'px_height'
# Replace values less than 217 with 217 in the specified column using numpy
df[column_name] = np.where(df[column_name] < 217, 217, df[column_name])

column_name = "sc_w"
# Replace values less than 2.82 with 2.82 in the specified column using numpy
df[column_name] = np.where(df[column_name] < 2.82, 2.82, df[column_name])
```

```
X_train = df.drop(columns=['m_dep', 'talk_time', 'clock_speed', 'n_cores', 'pc',  
'blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi'], axis=1)  
y_train = df['price_range']  
  
ids = df2['id']  
X_test = df2.drop(columns=['id', 'm_dep', 'talk_time', 'clock_speed', 'n_cores',  
'pc', 'blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi'], axis=1)
```

5. Referensi

- Chatterjee, S. (2019, November 24). *Use Naive Bayes Algorithm for Categorical and Numerical Data Classification*. Medium. Retrieved November 26, 2023, from <https://medium.com/analytics-vidhya/use-naive-bayes-algorithm-for-categorical-and-numerical-data-classification-935d90ab273f>
- Lembaga Penelitian dan Pengabdian Masyarakat Universitas Medan Area (2023, February 16). *Algoritma K-Nearest Neighbors (KNN) – Pengertian dan Penerapan*. LP2M UMA. Retrieved November 26, 2023, from <https://lp2m.uma.ac.id/2023/02/16/algoritma-k-nearest-neighbors-knn-pengertian-dan-penerapan/>
- Maulidevi, N. U. (2023, November 6). *Supervised Learning: K-Nearest Neighbors* [PowerPoint Slides]. Edunex IF3170 Inteligensi Buatan Teknik Informatika ITB 2023. https://cdn-edunex.itb.ac.id/53145-Artificial-Intelligence-Parallel-Class/210071-Supervised-Learning/90133-Supervised-Learning/1699250331293_IF3170_Materi09_Seg01_AI-kNN.pdf
- Maulidevi, N. U. (2023, November 6). *Supervised Learning: Naive Bayes* [PowerPoint Slides]. Edunex IF3170 Inteligensi Buatan Teknik Informatika ITB 2023. https://cdn-edunex.itb.ac.id/53145-Artificial-Intelligence-Parallel-Class/210071-Supervised-Learning/90133-Supervised-Learning/1699250380758_IF3170_Materi09_Seg02_AI-NaiveBayes.pdf

6. Pembagian Pekerjaan

Berikut merupakan pembagian pekerjaan pada Tugas Besar 1 IF3170 Inteligensi Buatan.

Tabel 6.1 Pembagian Pekerjaan

NIM	Nama	Pekerjaan
13521057	Hosea Nathanael Abetnego	Naive-Bayes
13521059	Arleen Chrysanthia Gunardi	KNN
13521127	Marcel Ryan Antony	Naive-Bayes
13521145	Kenneth Dave Bahana	KNN

7. Lampiran

Repositori GitHub: https://github.com/MarcelRyan/Tubes2_AI_haiyaaa